z

Automatic Discovery of Abnormal Values in Large Textual Databases

PETER CHRISTEN, The Australian National University
ROSS W. GAYLER, Veda
KHOI-NGUYEN TRAN, The Australian National University
JEFFREY FISHER, The Australian National University
DINUSHA VATSALAN, The Australian National University

Textual databases are ubiquitous in many application domains. Examples of textual data range from names and addresses of customers to social media posts and bibliographic records. With online services, individuals are increasingly required to enter their personal details for example when purchasing products online or registering for government services, while many social network and e-Commerce sites allow users to post short comments. Many online sites leave open the possibility for people to enter unintended or malicious abnormal values, such as names with errors, bogus values, profane comments, or random character sequences. In other applications, such as online bibliographic databases or comparative online shopping sites, databases are increasingly populated in (semi-) automatic ways through web crawls. This practice can result in low quality data being added automatically into a database. In this paper we develop three techniques to automatically discover abnormal (unexpected or unusual) values in large textual databases. Following recent work in categorical outlier detection, our assumption is that 'normal' values are those that occur frequently in a database, while an individual abnormal value is rare. Our techniques are unsupervised and address the challenge of discovering abnormal values as an outlier detection problem. Our first technique is a basic but efficient q-gram set based technique, the second is based on a probabilistic language model, and the third employs morphological word features to train a one-class support vector machine classifier. Our aim is to investigate and develop techniques that are fast, efficient, and automatic. The output of our techniques can help in the development of rule-based data cleaning and information extraction systems, or be used as training data for further supervised data cleaning procedures. We evaluate our techniques on four large real-world data sets from different domains: two US voter registration databases containing personal details, the 2013 KDD Cup data set of bibliographic records, and the SNAP Memetracker data set of phrases from social networking sites. Our results show that our techniques can efficiently and automatically discover abnormal textual values, allowing an organization to conduct efficient data exploration, and improve the quality of their textual databases without the need of requiring explicit training data.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Textual Databases*; H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.5.4 [**Pattern Recognition**]: Applications—*Text Processing*

General Terms: Algorithms, Experiments

Additional Key Words and Phrases: String databases, data quality, probabilistic language model, support vector machine, one-class classifier, out-of-vocabulary, word features, outlier detection.

ACM Reference Format:
Peter Christen, Ross W. Gayler, Khoi-Nguyen Tran, Jeffrey Fisher and Dinusha Vatsalan, 2016. Discovering Abnormal Values in Large Textual Databases. *ACM J. Data Inform. Quality* x, y, Article z (2016), 27 pages.
DOI : http://dx.doi.org/10.1145/0000000.0000000

This work is funded by the Australian Research Council (ARC), Veda, and Funnelback Pty. Ltd., under Linkage Project LP100200079.
Author's addresses: P. Christen, K.-N. Tran, J. Fisher, and D. Vatsalan, The Australian National University, Research School of Computer Science, Canberra ACT 0200, Australia, email: {peter.christen, khoi-nguyen.tran, jeffrey.fisher, dinusha.vatsalan}@anu.edu.au; R.W. Gayler, Melbourne, Australia, email: r.gayler@gmail.com.

DOI : http://dx.doi.org/10.1145/0000000.0000000

Table I: Selected examples of discovered values with high abnormality scores (some slightly shortened for presentation) from our experimental data sets, with one row per technique (**BSI / PLM / SVM**) as presented in Sections 4 to 6. As can be seen, our techniques can identify various types of abnormal values, including values that contain small typos only, contain some rare abbreviation or mistaken words or tokens, are rare names from another culture, or are obvious completely unexpected and wrong values for a given attribute.

Data set / attribute	Technique	Example abnormal values
2013 KDD Cup / Paper title	**BSI**	Euporean Community Euro 0.4394 0.4335 0.4572 0.4599 0.477
	PLM	R,S-4-(4'-METHOXYBENZYLIDENEAMINO) PHENYLALANINE
	SVM	The CH3 3pz 2A2
2013 KDD Cup / Author name	**BSI**	Henri Epstein ORF RID="a3">
	PLM	PALAISEAU CEDEX
	SVM	h-z. hu
North Carolina Voter Registration / First name	**BSI**	"we18" 82"n24 #894 w "46un2
	PLM	062"
	SVM	za'heem
North Carolina Voter Registration / Street address	**BSI**	3' fernwood ct
	PLM	6 atka ct #b3
	SVM	509 m & m ln
SNAP Memetracker / Phrases	**BSI**	tag blogger com 1999 blog-31636059 post-3432464819012682047
	PLM	1e 2f 3g 4h 5i
	SVM	wild updated 2009-03-31 16 44 26 gmt

1. INTRODUCTION

Textual data, such as personal details, social networking posts, or bibliographic records, are being collected in many application areas. With the progress of Web technologies it is becoming more common for end users to be entering such data by themselves, or for such data to be collected and integrated from diverse internal and external sources in (semi-) automatic ways.

For online business and government services, for example, the end users are not highly trained employees of an organization, and they often have no responsibility for the accuracy of the data created from their system interactions. The providers of the user interface may have no interest in catering for all the use cases (such as exploring system functionality or system testing). Consequently, end users may enter inappropriate values (such as 'John Test 123', '999 Unknown Street', 'fefsl6de8', or '123-456-7890'), because a user is for example not serious about actually ordering anything from an online store, or a user interface does not allow for testing.

On social networking sites and online stores, users are commonly allowed to post comments (for example to discuss product offers) and to reply to posts by others. This leaves open the possibility of malicious and profane comments, as well as random gibberish. Being able to efficiently identify such free-format text comments will allow inappropriate or random content to be detected and removed before it is stored in a database or made visible to subsequent visitors to the site.

In other applications, when data collected from external sources are automatically integrated into a database or data warehouse, it is possible that inappropriate and abnormal values are added to the database because manual oversight of the collection and integration process is commonly not practical. Example applications are online bibliographic databases, such as *Google Scholar* or *Microsoft Academic Search*, which crawl the Web to gather academic publications to be integrated and matched such that all publications by a certain author are correctly attributed to them.

Consequently, in all these applications an underlying database can be polluted with abnormal and inappropriate textual values. Even if these are quite rare they interfere with the interpretation and analysis of the database and any other processes that depend upon its quality. Table I shows a small set of real examples of abnormal values discovered by our three techniques from the data sets we use in our experiments in Section 7 (more examples are provided in Section 8).

In this work we aim to automatically discover abnormal textual values by calculating a score for abnormality for values that occur rarely in a database. Following recent work in categorical outlier detection by Wu and Wang [2013], we use the frequency of values in a database as a heuristic proxy for normality, where we assume that frequent values are more likely to be normal and rare values

are more likely to be abnormal. Because textual attributes commonly exhibit a skewed frequency distribution (a few common values and a long tail of rare values) [Witten et al. 1999], the majority of rare values are however normal. We learn models of 'normality' from frequent values and apply those models to rare values. We rank rare values according to their abnormality calculated as a score on how different they are from the learned models. Our approaches are unsupervised and allow an improvement of data quality in textual databases without the need of manually prepared cleaning rules or labeled training data.

We note that the distinction between normal and abnormal is rather subtle. We distinguish between rarity in the population and in a database. Values that are common in a database will necessarily be common in the population, but values that are equally rare in a database (e.g. occur only once) will have a range of rarity in the population because of sampling variability (i.e. some values that are relatively more common in the population will not appear in a database, and some values that appear in a database will be relatively more rare in the population). Our aim is to estimate models of population rarity from the observed frequencies in a database, and identify observed values with very low estimated probability as anomalous.

We develop techniques that allow progressively more complex ways of identifying abnormal values. We expect simple techniques are more likely to be used in actual real-world data cleaning systems. We investigate how much can be achieved with simple methods, and this also provides us with a baseline for comparison with more complex techniques. We contribute three different techniques (summarized below) of increasing computational complexity, and compare the abnormal values they identify.

We believe our work addresses an important real problem of data quality in textual databases, where identifying abnormal values without manual intervention is a crucial initial step to improve data quality. Many data cleaning systems require manual input in the form of patterns (like regular expressions) or rules to guide the cleaning process. However, developing such rules is challenging even for experienced domain experts. Our techniques are a way to bootstrap the data cleaning process by providing abnormal values that can be used to (manually) develop rules, or as labeled examples of abnormal values for supervised classification and outlier detection techniques.

Our work is aimed at databases where textual values are either entered manually, or where data are collected in (semi-) automatic ways. These are both increasingly common practices in many applications for both businesses and governments, as services are provided online and as organizations aim to collect external data that are relevant for their processes.

Contributions: We contribute three different unsupervised techniques for identifying abnormal values in large textual databases. A first basic technique (named **BSI**) calculates the intersection of sets of character q-grams (sub-strings of length q) to find q-grams that only occur in rare values. The second technique (named **PLM**) is based on a probabilistic language model and calculates the likelihood that a value is being generated by a model trained on frequent values, taking dependencies between q-grams into consideration. The third technique (named **SVM**) employs morphological word features and a one-class support vector machine classifier to learn which features distinguish normal from abnormal values. We evaluate these techniques on four large real-world data sets: two voter registration databases from the US state of North Carolina, one of which appears to contain significantly more corrupted values than the other database; the *2013 KDD Cup* data set which contains bibliographic data collected by *Microsoft Academic Search*; and finally a large data set of phrases from *Memetracker* as collected by the *Stanford SNAP* project [Leskovec and Krevl 2014]. We provide examples of abnormal values discovered by our three proposed techniques, and analyze and discuss their efficiency and effectiveness. While we would expect high data quality especially in the voter registration databases, our experiments show this is not the case.

Outline: In the following section we review related work. In Section 3 we provide an overview of our proposed approach, and in Sections 4 to 6 we describe our three techniques in detail. In Section 7 we provide details of our experimental evaluation, and we present and discuss results in Section 8. We finalize our work with conclusions and an outlook to future work in Section 9.

2. RELATED WORK

Discovering abnormal, unusual, or unexpected values has been investigated in various application domains using a variety of approaches. Most of these approaches are based on some sort of unsupervised anomaly detection, because either abnormal values are not available for training a fully supervised approach, or their characteristics are unknown. In the following we review work in anomaly detection in categorical and textual data, data cleaning of string values, one-class classification, modeling of out-of-vocabulary words, and rule-based data cleaning.

While most work in anomaly detection has concentrated on numerical data or data of mixed types [Chandola et al. 2009; Hodge and Austin 2004], several techniques have recently been proposed that are aimed at categorical and textual data. Chandola et al. [2012] proposed a taxonomy of anomaly detection algorithms in discrete sequences and characterized existing techniques along this taxonomy. The taxonomy consists of three main problem formulations: (1) sequence-based anomaly detection, where the task is to detect anomalous sequences from a database of test sequences, (2) contiguous sub-sequence-based anomaly detection, where the task is to detect anomalous contiguous sub-sequences within a long sequence, and (3) pattern frequency-based anomaly detection, which aims to detect (possibly non-contiguous) patterns in a test sequence with anomalous frequencies of occurrence.

According to this taxonomy, our proposed approaches belong to the problem of *sequence-based anomaly detection* for detecting anomalous sequences (in our case the sequences are textual attribute values) from a database of test sequences. Among the two variants of semi-supervised and unsupervised anomaly detection of this type of problem formulation, we focus on the later where the task is to assign an anomaly score to each sequence in a set of sequences with respect to the rest of sequences. We next review related work following the taxonomy proposed by Chandola et al. [2012].

2.1. Sequence-Based Anomaly Detection Techniques

Das and Schneider [2007] aimed to find individual anomalous records that contain unusual combinations of categorical attribute values using conditional probability tests across attributes. Das et al. [2008] extended this work to consider anomalous groups of records with categorical attributes. McFowland et al. [2013] proposed the fast generalized subset scan algorithm which aims to detect anomalous patterns in categorical data within subsets of both records and attributes. A Bayesian model is learned from training data, and for each attribute value in each record the algorithm calculates the likelihood that the value is generated by that model. Subsets of records and attributes are then ranked according to how unlikely they were generated by the model, assigning anomalous subsets with high scores. This approach outperformed the earlier work by Das et al.

An unsupervised approach based on a combination of the two measures entropy and total correlation was recently developed by Wu and Wang [2013]. The authors proposed a new outlier factor and proved that, based on these two measures, the likelihood of an attribute value to be an outlier decreases monotonically with an increase in the frequency of the value. This is in line with our assumption that frequent values are less likely to be abnormal values compared to rare values.

Similar to our techniques, all these approaches belong to the problem of sequence-based anomaly detection in the taxonomy proposed by Chandola et al. [2012]. However, these approaches define outliers as being unusual across several categorical attributes, while we aim to detect unusual values within an individual attribute.

2.2. Contiguous Sub-Sequence-Based Anomaly Detection Techniques

Guthrie et al. [2007] proposed five different methods for unsupervised anomalous segment detection in text documents based on more than 200 stylistic features to characterize text segments. The output of this characterization is a feature vector for each text segment and a corresponding feature vector for its complement (i.e. the union of the remaining segments). Five measures were evaluated to compute the difference between text segments in order to identify anomalous segments. While this approach requires longer text sequences, our techniques work on short strings as well.

An efficient tree-based approach for detecting unusual sub-strings in long biological sequences, such as DNA sequences, was proposed by Apostolico et al. [2000]. A statistical suffix-tree index for a string is built that efficiently annotates the count frequencies, expected values, and variances of sub-strings based on probabilistic assumptions. A statistical method is then used to calculate an anomaly score based on the number of occurrences of each sub-string to efficiently identify the sub-strings in the index tree that exhibit an anomalous pattern.

Both of the approaches by Guthrie et al. [2007] and Apostolico et al. [2000] addressed the contiguous sub-sequence-based anomaly detection problem, which is a different problem compared to ours as characterized by the taxonomy by Chandola et al. [2012].

2.3. One-class Classification and Feature-Based Techniques

One-class classification has been used in applications ranging from recognizing hand-written digits to spam email detection [Khan and Madden 2010]. These techniques are suitable when it is either impossible or too expensive to obtain or generate training data from both a positive or negative class, or where one of the two classes is ill defined.

On Wikipedia, detection of rare text values has contributed to maintaining the quality of articles by predicting which articles contain flaws. Flaws range from minor misspellings to overall poorly written articles. Anderka et al. [2011] used a variety of features on article content and structure, links between articles, and edit history, with a density-based one-class classifier to identify different quality flaws and classify articles as being normal or outlier. Incoming article edits can then be screened for potential flaws for an editor to manually inspect.

In computational linguistics, Müller and Schütze [2011] used word features to model out-of-vocabulary words and cluster rare values into morphological classes (i.e. words with similar characteristics) using a supervised approach. The authors proposed a set of novel features (similar to the ones shown in Table II) and evaluated their approach on the Wall Street Journal (WSJ) corpus of over 50 millions words. Results showed improved perplexity (described in Section 5) compared to the state-of-the-art trigram-based Kneser-Ney model [Müller and Schütze 2011].

Bikel et al. [1999] applied a hidden Markov model (HMM) trained on word features to learn what distinguishes names, dates, and times from general text in the context of named entity recognition. The features used are similar to the ones we use in our one-class SVM technique, as shown in Table II. Similar to Müller and Schütze [2011], the authors also used the WSJ corpus, as well as a corpus of Spanish news articles, for experimental evaluation. Their results showed that texts with mixed upper- and lower-case words can lead to better recognition results compared to texts with words in one case only.

Both the approaches by Müller and Schütze [2011] and Bikel et al. [1999] addressed the different problems of part-of-speech tagging or out-of-vocabulary word identification, while we use word features along with a one-class SVM classification for anomaly detection in textual databases.

2.4. Data Cleaning Applications

In the context of business and enterprise applications, textual data are becoming more widespread as organizations increasingly communicate with their customers online, provide ways for commenting on products and services, and engage in online social networking. Efficient cleaning and analyzing of massive textual databases therefore become crucial. One aim of cleaning is to identify and remove meaningless or inappropriate content. Information extraction is another key component of the text analytics pipeline, where one aspect is the annotation of text, for example mentions of consumer products or customers' positive or negative comments. Such annotations are often based on rules. While traditionally data cleaning and annotation rules have been crafted manually, which is a time and labor intensive process, techniques to automate this have recently been proposed.

The relative rarity of specific errors or misspellings has been used by Ciszak [2008] in two unsupervised techniques for creating validation rules for data pre-processing. The first technique is context independent and considers each attribute in isolation. The values for each attribute are clustered using a string distance such as edit distance [Christen 2012]. After clustering, the most frequently

occurring value in a cluster is treated as the correct one, while the other values are treated as errors or misspellings. The second technique is similar but also incorporates dependencies between the attributes by using association mining, which allows for specific values that are rare overall, but in the context of other attributes could be normal, to be identified as normal. The techniques developed by Ciszak [2008] are designed for well defined domains with small numbers of unique values, such as city names, and where the number of records is sufficiently large that each correct attribute value likely occurs many times. In contrast, in the problem we consider the universe of possible strings is effectively open-ended and individual normal values are not guaranteed to occur frequently.

Mazeika and Böhlen [2006] proposed a string clustering algorithm aimed at grouping similar strings. Their proximity graph cleaning (PGC) approach does not require any parameters but finds the boundaries between clusters by exploring the neighborhood of strings. All strings in the same cluster are replaced with the most frequent spelling in the cluster (which is assumed to be the correct spelling). The major disadvantage of this approach is its high computational complexity, which was addressed by Kazimianec and Augsten [2011] who developed efficient pruning methods for the PGC algorithm. While this approach works well to correct misspellings, it will not detect abnormal values that are rare and have unique characteristics because such values will likely not be clustered with any normal values, but rather with other abnormal values.

A related problem in text cleaning and processing is text normalization, where the aim is to map out-of-vocabulary words to their most similar in-vocabulary (i.e. known and correct) words. Recent work by Zhang et al. [2013] aimed to solve this problem within the context of a language parser. A normalization graph is generated over a sentence that contains unknown words, and based on training data and a small set of domain specific generators, unknown words are substituted with the most likely known words, resulting in a grammatically correct sentence. This work assumes that text consists of a sequence of tokens where only one or a small number of tokens are unknown words. In contrast, our techniques do not make any assumption about the content of the textual values in an attribute, and as our experimental results show (in Section 8), they work well with both short and single word values, as well as with longer multi-word values.

Many of the discussed data cleaning techniques, and prototypes such as *CerFix* [Fan et al. 2011] and *Cleanix* [Wang et al. 2014] that are built on top of them, require an initial set of rules or examples of abnormal values to be identified in a database to bootstrap the rule development or rule learning process. Our techniques, introduced in Section 3, automatically detect abnormal text values and can therefore be used as a precursor for such rule- and learning-based data cleaning and information extraction approaches.

2.5. Pattern-Based Anomaly Detection Techniques

IBM's *SystemT* [Li et al. 2011] supports the development of rules based on a declarative rule language which is similar to SQL, allowing rule developers to concentrate on what to extract and not how. To facilitate rule development, supervised techniques that learn regular expressions from training data have also been developed. Li et al. [2008] used both labeled examples as well as a set of initial regular expressions to learn rules that are better than the initial set by systematically exploring possible regular expression transformations (i.e. variations of the initial rules). A similar approach was presented by Liu et al. [2010] who generated and ranked refined rules based on data provenance information (i.e. how values have changed over time). All these techniques require a way of identifying abnormal values that can be used to inform rule development or learning. Such values can be automatically identified using our techniques, as presented in the following sections.

3. OVERVIEW OF AUTOMATIC TECHNIQUES TO DISCOVER ABNORMAL VALUES

We first assume that no training data with 'normal' and 'abnormal' textual values are available. Instead, our idea to discover abnormal values is based on the hypothesis that values that occur relatively frequently in a database are more likely to be normal compared to individual abnormal values that generally occur only once or a few times [Wu and Wang 2013]. A value occurring only once in a database is known as a *singleton* [Jurafsky and Martin 2008]. However, as textual attributes

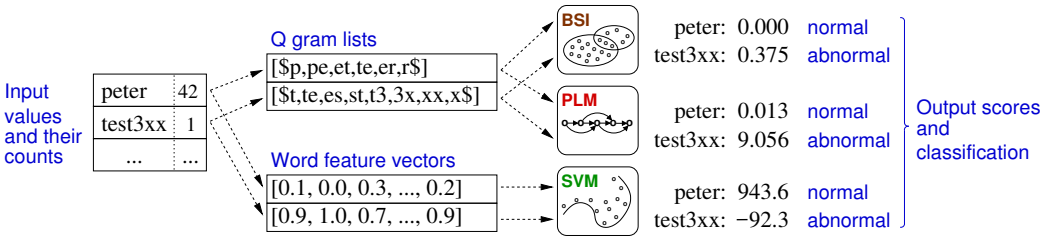


Fig. 1: Overview of our three techniques to automatically discover abnormal textual values. Details of these techniques are provided in Sections 4 to 6. The input is a set of textual values and their counts (frequencies) of occurrences in a database, and the output are scores of their abnormality (note the different scores for the three techniques). The '$' character symbolizes the padding of q-grams at the beginning and end of attribute values.

ALGORITHM 1: Overview of automatic discovery of abnormal values

Input: $\mathbf{C}$: scoring classifier to be trained and employed.
V: set of attribute values v_i and their counts c_i.
m: minimum count of values in V to be selected as assumed normal values.
t: proportion of assumed normal values to be used for training of classifier $\mathbf{C}$, with $0 < t < 1$.
Output: A: list of abnormal values sorted according to their abnormality scores.

1: $T = \{v_i \in V : c_i \geq m\}$; // Select values that occur at least m times.
2: $T_t = \{v_i \in T : |T_t| = t \times |T|\}$ randomly selected; // Split T into a training set and
3: $T_s = \{v_j \in T : v_j \notin T_t\}$; // a testing set.
4: $S = \{v_j \in V : c_j < m\}$; // Candidate set of rare, potentially abnormal, values.
5: $\mathbf{C}.train(T_t)$; // Train the classifier.
6: $trnp = \mathbf{C}.test(T_s)$; // Assess classifier performance.
7: $N, A = \mathbf{C}.employ(S)$; // Classify rare values in S into normal and abnormal.
8: $tsap = |A|/(|A| + |N|)$ // Calculate percentage of abnormal values in S.
9: $rank(A)$ // Sort according to abnormality scores of $v_j \in A$.

often follow a Zipf-like distribution [Witten et al. 1999], many rare values and singletons will be legitimate normal values. Our challenge is to distinguish abnormal rare values from normal ones.

We define an abnormal textual attribute value as *a value that has different characteristics compared to the characteristics of the majority of frequent values of a given database attribute.* For example, normal last names would be single or hyphenated words that only contain letters; while normal telephone numbers would only contain digits, hyphens, brackets, and maybe spaces, have a certain minimum and maximum length, and also follow a certain grouping of digits.

In the following sections we propose and describe three different techniques to automatically discover abnormal textual values. Figure 1 provides an overview of our techniques. Two of them convert attribute values into character q-grams and either use a basic set-intersection (named **BSI**) or a probabilistic language model (named **PLM**) based approach. Different from the first two, the third technique extracts morphological word features from attribute values which are then used to train a one-class support vector machine classifier (named **SVM**).

Algorithm 1 illustrates the main steps common to all three techniques, using the following assumptions and notation. We assume a database with attributes that contain textual values, such as names and addresses of customers; titles, authors, and venues of scientific publications; or comments posted by users on social networks or online stores. Without loss of generality, our notation assumes only a single attribute. For a given attribute, we denote the set of all its unique values with V, with $v_i \in V$ being one specific attribute value, and c_i the count (frequency) of how often v_i occurs in this attribute in the database (i.e. the number of records in the database where a given

attribute value is v_i). If a database contains n records, we assume that $\sum_i c_i = n$. We do not include empty or missing values in V.

Our hypothesis is that values v_i that occur sufficiently often in a database, i.e. they have a sufficiently high value c_i, with $c_i > 1$, are extremely likely to be normal values. On the other hand, values that only occur rarely, potentially $c_i = 1$ (singletons), are relatively more likely to be the abnormal values we aim to discover. We assume that the generative processes for normal and abnormal values are different, where each specific abnormal value is unlikely to occur frequently in an attribute.

Based on these assumptions, our aim is to have a model (in the form of a scoring classifier $\mathbf{C}$) that learns the properties of normal values from a corpus of almost purely normal values. We therefore only select values with high counts for training. Specifically, as shown in line 1 of Algorithm 1, we generate a training set $T \subset V$, with m ($m > 1$) the minimum count c_i for a value $v_i \in V$ to be an element of T. These are the assumed normal values we use to train and evaluate our classifier $\mathbf{C}$. We randomly split T in lines 2 and 3 into a training set T_t and a test set T_s using the splitting parameter t ($0 < t < 1$) which is the proportion of values in T to be used for training $\mathbf{C}$. In line 4 we generate the set S of values from V that are not in T (these are the values we will classify and rank in lines 7 and 8 to obtain their abnormality scores). We then train the classifier $\mathbf{C}$ on T_t in line 5 and evaluate it on T_s in line 6. We calculate a *training normal percentage*, $trnp$, defined as the percentage of values in T_s that are classified as being normal. This gives us a measure of how well a classifier is able to correctly label values from the training set as normal.

The values in S are then split using the trained scoring classifier $\mathbf{C}$ in line 7 into the set N of normal and the set A of abnormal values. We calculate a *testing abnormal percentage*, $tsap$, defined as the percentage of values in S that are labeled as being abnormal (i.e. are in A). The values $v_j \in A$ are then sorted in line 9 according to their abnormality scores to identify the most abnormal values. Details of these steps for the different techniques are provided in the following three sections.

Note that in a practical application, if the labeled abnormal values are flagged for manual review, another threshold will most likely be set according to the resources available, such that only the most abnormal values in A (according to their abnormality score) will be manually assessed. We also note that the $trnp$ and $tsap$ are calculated with respect to the split of V into T and S which we use as a proxy for normal and abnormal values, respectively, rather than a true (supervised) split of V into truly normal and truly abnormal values which is almost never going to be available in practical applications.

4. BASIC SET-INTERSECTION TECHNIQUE

This simple but very efficient technique, named **BSI** uses character q-grams (sub-strings of length q characters, with $q \geq 1$) extracted from values in V, and calculates how many q-grams in a potentially abnormal value $v_j \in S$ have not occurred in the training set T_t. The more q-grams in v_j that have not been seen in T_t the more unusual the value v_j is. A suitable value of q needs to be chosen based on the lengths of the values in V [Christen 2012].

We use a function $Q_j = qgram(v_j, q)$ which extracts the list Q_j of character q-grams from a value v_j by using a sliding window approach [Christen 2012]. For $q > 1$ we pad a value with $q - 1$ special characters to distinguish the q-grams at the beginning and end of a value. For example, with $q = 2$ (bigrams), the q-gram list extracted from peter is $qgram(\text{peter}, 2)$ = [p, pe, et, te, er, r], where '$' is the padding character. Training the **BSI** classifier in line 5 of Algorithm 1 consists of generating the set Q_T of all unique q-grams that occur in any value in T_t:

$$Q_T = \{q_k \in qgram(v_i, q) : v_i \in T_t\}. \quad (1)$$

Testing the classifier in line 6 is done by calculating for each value $v_j \in T_s$ the number of q-grams that do not occur in Q_T, and based on this to calculate the *abnormality* score a_{BSI} of unknown

q-grams for v_j:

$$a_{BSI}(v_j) = \frac{|Q_j| - |Q_T \cap Q_j|}{|Q_j|}, \tag{2}$$

where Q_j is the list of q-grams extracted from v_j. The larger $a_{BSI}(v_j)$ is, the more abnormal v_j is.

The *training normal percentage*, $trnp$ (line 6 in Algorithm 1), of the **BSI** technique can now be calculated as the ratio of values in $v_j \in T_s$ that have an abnormality score $a_{BSI}(v_j)$ of 0 (these are the values where all their q-grams have occurred in the training set T_t):

$$trnp_{BSI}(T_s) = \frac{|\{v_j \in T_s : a_{BSI}(v_j) = 0\}|}{|T_s|}. \tag{3}$$

When employing the **BSI** classifier (line 7 in Algorithm 1) we calculate $a_{BSI}(v_j)$ for each $v_j \in S$, and add the v_j that have $a_{BSI}(v_j) > 0$ to the set A of abnormal values, and those with $a_{BSI}(v_j) = 0$ to the set N of normal values.

While being very simple, the **BSI** technique has several advantages over the other two techniques described next. First, it is very fast as it only requires a set intersection. Assuming a $O(1)$ set test-membership operation, then calculating Equation 2 is of $O(|Q_T|)$. In a worst case, assuming each $v \in V$ has its own set of unique q-grams, this becomes $O(|V|)$. However, in practice, the size of Q_T grows sub-linearly with the size of V which itself grows sub-linearly with the size of the database it is extracted from.

5. PROBABILISTIC LANGUAGE MODEL

The **BSI** approach only counts the proportion of q-grams in a value that do not occur in the training set. It does not take into account the counts of occurrences of these q-grams, nor the dependencies between consecutive q-grams. Such information can help identify values that are very unlikely to be normal even if all of their q-grams occur in the training set. An abnormal value can for example be one that is made of a sequence of q-grams that each occurs only once in the training set, or where the specific ordering of q-grams is highly unusual.

Probabilistic language models [Jurafsky and Martin 2008; Manning and Schütze 1999] are used in natural language processing for applications such as machine translation, speech recognition, part-of-speech tagging, or handwriting recognition. They assign a relative probability to different phrases (i.e. sequences of words) to identify which phrase is the most likely based on a model trained on a large corpus of phrases. Such models take both frequency and ordering information of words into account by calculating the likelihood that a certain sequence of words has been generated by a model trained on a corpus of training values.

In this section we use the ideas of probabilistic language models and describe the **PLM** technique that extends the simple **BSI** approach by taking both the counts of q-grams and the dependencies between them into account. As with the **BSI** technique, the elements of our sequences are q-grams rather than words. Specifically, we estimate the probability that a q-gram q_k in an attribute value occurs after a consecutive sequence of certain other q-grams $q_{k-(n-1)}, \ldots, q_{k-1}$ as:

$$p(q_k | q_{k-(n-1)}, \ldots, q_{k-1}) = \frac{count(q_{k-(n-1)}, \ldots, q_{k-1}, q_k)}{count(q_{k-(n-1)}, \ldots, q_{k-1})}, \tag{4}$$

where n is the order of the model [Jurafsky and Martin 2008] assuming the Markov property that the current q-gram q_k only depends on the previous consecutive $n-1$ q-grams. These models are commonly known as *unigram* ($n = 1$), *bigram* ($n = 2$), or *trigram* ($n = 3$) models. The function $count(\cdot)$ returns the number of times the function argument occurs in values in a data set. In the special case $n = 1$, the probability of a q-gram q_k is estimated based only on its frequency of occurrence as:

$$p(q_k) = \frac{count(q_k)}{\sum_{q_k \in Q} count(q_k)}, \tag{5}$$

where Q is the set of all unique q-grams extracted from values in a data set.

The overall probability for a value v is then approximated as the product of the probabilities calculated using Equation 4 or 5 over the sequence of q-grams in v (where for q-gram lengths $q > 1$ values v are padded as described in Section 4) as:

$$p(v) = \begin{cases} \prod_{q_k \in v} p(q_k | q_{k-(n-1)}, \ldots, q_{k-1}) & \text{if } n > 1, \\ \prod_{q_k \in v} p(q_k) & \text{if } n = 1. \end{cases} \quad (6)$$

In our application, the values v are often short, and thus only contain a few q-grams. In the experiments in Section 7 we will therefore only use small values for q and n ranging from 1 to 3.

The numerator and denominator probabilities in Equations 4 and 5 can be very small for rare values and the counts can therefore be noisy approximations of the actual population probabilities. Certain q-grams or q-gram tuples might also only occur in the set S of potentially abnormal values but not in the training set T_t, leading to zero counts returned by the $count(\cdot)$ function during testing and use of the **PLM** classifier. This issue, known as the *zero-probability problem*, is common in natural language modeling due to the sparseness and open-world assumption of natural language, where never before seen sequences of words (that are however grammatically correct) can occur [Jurafsky and Martin 2008]. In the domain of detecting abnormal values the zero-probability problem is even more prominent. In essence we are trying to detect values that are predicted to be so rare that they should not occur, but they actually do occur once or a few times in a database. That is, we are trying to estimate predicted and actual probabilities of values that are so rare that they are on the edge of observability, and where the observations are very noisy.

To overcome the noisiness and sparseness of rare q-grams and q-gram tuples, we apply *smoothing* [Chen and Goodman 1999] to modify the potentially poor probability estimates for rare values in Equation 6. Various smoothing techniques have been developed for probabilistic language models [Chen and Goodman 1999], and we adapt three for our **PLM** classifier.

— **Laplace smoothing:** In this technique [Jurafsky and Martin 2008], we add 1 to all q-gram and q-gram tuple counts (as returned by the $count(\cdot)$ function) both during training and testing of the **PLM** classifier, and therefore set the counts of those q-grams or q-gram tuples that are never seen during training to 1. As in natural language processing, the drawback with Laplace smoothing is that potentially too much probability mass is assigned to unknown q-grams or q-gram tuples (that only occur in the set S of rare values) if their numbers are large compared to the known ones (i.e. those that occur in the training set T_t).

— **Absolute discount smoothing:** This technique aims to overcome the drawback of Laplace smoothing by only assigning a small fixed probability to all unknown q-grams and q-gram tuples [Borkar et al. 2001; Chen and Goodman 1999]. Specifically, for q-gram tuples we calculate the value $x = 1/(s_Q + n_Q)$, where s_Q is the sum of counts of all q-gram tuples from the training set T_t (i.e. the sum of all denominators in Equation 4), while n_Q is the total number of unique q-gram tuples that occur in any attribute values in V. The probability in Equation 4 of each q-gram tuple in the training set T_t is then reduced by x, while we assign the probability $p(q_k) = n_T x/(n_Q - n_T)$ to each q-gram tuple that only occurs in the test set S, where n_T is the number of q-gram tuples that occur in the training set T_t only (known q-gram tuples). For $n = 1$, we similarly discount the probabilities for known q-grams in Equation 5.

— **Simple Good-Turing:** The intuition behind this technique is to use the count of values seen once in a data set to estimate the count of values not seen in that data set [Jurafsky and Martin 2008]. This technique requires us to calculate the frequencies of frequencies, i.e. how many q-gram tuples occur a certain number of times in values in V. If we denote with N_c the number of q-gram tuples that occur c times, our aim is to estimate N_0, the number of q-gram tuples we have not seen ($c = 0$). The original Good-Turing approach replaces a count c with the estimated count $c^* = (c+1)N_{c+1}/N_c$. The approach however does not use this equation to directly estimate the smoothed count for N_0, rather it calculates $p(N_0) = N_1/N$, where N is the total number of all q-gram tuples seen, $N = \sum N_i$ with $i > 0$. Due to the potential occurrence of zeros in the N_c (i.e. no

q-gram tuples occur c times for certain values of $c > 0$, which is common for larger c), the above equation can be undefined for certain values of c. Simple Good-Turing [Gale and Sampson 1995] overcomes this issue by replacing the values N_c with values calculated using a simple linear regression which fits N_c to c in log space as $log(N_c) = a + b\ log(c)$. Often this smoothing of the estimated counts c^* is only applied for small values of c, while for larger counts the original values, as calculated from a data set, are used. In our work we limit smoothing to values $c \leq 10$ [Jurafsky and Martin 2008].

After smoothing has been applied, our **PLM** language model (classifier **C** in Algorithm 1) calculates the (smoothed) probabilities in Equation 6 for each training set value $v_i \in T_t$. The lower a $p(v_i)$ is the less likely v_i is being generated by the model, and the more likely v_i is abnormal.

In language modeling, the measure of *perplexity* is commonly used to evaluate how well a (long) sequence of words fits a trained language model [Jurafsky and Martin 2008], or to compare how well different models trained on the same data set perform. Based on cross-entropy, perplexity for a sequence of words $W = w_1, w_2, \ldots, w_l$ is calculated as:

$$P(W) = 2^{-\frac{1}{l}\sum_{i=1}^{l} log_2(p_i)}, \tag{7}$$

where p_i is the probability that word w_i occurs as calculated by the language model. Perplexity is a measure that normalizes for the length l of a sequence. A low perplexity $P(W)$ means the sequence W fits the trained language model well, while a high $P(W)$ means that the language model is not able to fit the sequence well, because the sequence is not consistent with the sequences the model has been trained on.

For our **PLM** classifier, instead of using sequences of words to calculate perplexity we use the sequences of q-grams extracted from values, and based on Equations 6 and 7 we calculate the perplexity of a value v as:

$$P(v) = 2^{-\frac{log_2(p(v))}{l_v}}, \tag{8}$$

where $l_v = |Q_v| + 1 - n$ is the number of q-gram tuples extracted from v, with n the order of the model and Q_v the q-gram list extracted from v as described in Section 4. We then calculate the maximum perplexity of all values in the training set T_t as:

$$P_{PLM} = max(P(v_i) : v_i \in T_t). \tag{9}$$

Any value $v_j \in S$ with a perplexity larger than P_{PLM} is less likely to be generated by the **PLM** language model trained on the set T_t and is classified as abnormal, while any value $v_j \in S$ with a perplexity equal to or lower than P_{PLM} is more likely to be generated by the **PLM** model than at least one value in T_t, and therefore will be classified as normal.

To test how well the **PLM** model fits the training set, we check how many values in T_s have a perplexity less than or equal to P_{PLM}. Ideally, all values in T_s should be classified as being normal, i.e. have a perplexity equal to or lower than P_{PLM}. In line with Equation 2, we calculate the abnormality score a_{PLM} as:

$$a_{PLM}(v_j) = \frac{P(v_j)}{P_{PLM}}, \tag{10}$$

where $P(v_j)$ is the perplexity of $v_j \in T_s$ calculated using Equation 8. A value of $a_{PLM}(v_j) > 1$ means v_j is classified as an abnormal value, otherwise it is classified as normal. Based on these a_{PLM} values, we then calculate the training normal percentage (line 6 in Algorithm 1) of the **PLM** technique as:

$$trnp_{PLM}(T_s) = \frac{|\{v_j \in T_s : a_{PLM}(v_j) \leq 1\}|}{|T_s|}. \tag{11}$$

This gives us the percentage of how many values in T_s are classified by the **PLM** language model as being normal values.

Table II: Word features used for the one-class SVM.

Name	Description	Name	Description
ValLen	Value length	**NumHyphen**	Number of hyphens (-)
NumWord	Number of words in value	**NumQuote**	Number of single quotes (‘)
AvrWLen	Average word length	**NumOther**	Number of other characters
MinWLen	Minimum word length	**NumUpp**	Number of uppercase letters
MaxWLen	Maximum word length	**NumLow**	Number of lowercase letters
NumCapWord	Number of capitalized words	**LonRepChr**	Longest repeated character (‘yyyy’ $\rightarrow 4$)
NumUppW	Number of uppercase only words	**LonRepSeq**	Longest repeated sequence (‘klnkln’ $\rightarrow 3$)
NumOneLetW	Number of single letter words	**LonRepSeqC**	Count of longest repeated sequence (‘abc1abc34abc67abcXabcQ’ $\rightarrow 5$)
NumOneDigW	Number of single digit words		
NumAlphaW	Number of alphabetic words	**LonIncSeq**	Longest increasing sequence (‘abcd’ $\rightarrow 4$)
NumDigW	Number of digit words	**LonDecSeq**	Longest decreasing sequence (‘6543’ $\rightarrow 4$)
NumAlnumW	Number of alphanumeric words	**RatLonShort**	Ratio of length of longest to shortest word
NumOthW	Number of other words (that include punctuation characters)	**RatUppLen**	Ratio of uppercase letters to length
		RatLetLen	Ratio of letters to length
NumLet	Number of letters	**RatDigLen**	Ratio of digits to length
NumDigit	Number of digits	**RatPunctLen**	Ratio of punctuations to length

The computational complexity of the **PLM** technique is linear in the number of unique q-gram tuples that occur in values $v_i \in V$. Their number crucially depends upon n, the order of the model.

6. WORD FEATURE ONE-CLASS CLASSIFIER

Based on ideas used in document classification [Manevitz and Yousef 2002], out-of-vocabulary modeling in computational linguistics [Müller and Schütze 2011], and quality flaw detection in Wikipedia [Anderka et al. 2011], we present an alternative approach (named **SVM**) to identify abnormal values by using morphological word features created from the values in V. Our hypothesis is that normal and abnormal values have different characteristics in such features.

For each value $v_i \in T_t$ we generate the thirty features shown in Table II, resulting in a vector $\mathbf{f}_i$ of numerical values describing v_i (integer feature values are normalized before being used such that the values in each feature are in the range $[0, 1]$). These features characterize the content of a value with regard to the number and length of words (white-space separated), the content of these words (e.g. letters, digits, punctuations, and upper- and lower-case letters, as well as the longest repeated sequences of characters), ratios of certain character types, and so on.

The one-class support vector machine (SVM) approach was originally proposed by Schölkopf et al. [2001] as a means to estimate the support of high-dimensional distributions. Named *one-class ν-SVM*, this approach maps data into a feature space such that the given data are separated from the origin with a maximum margin [Yang et al. 2010]. The algorithm learns a decision function d where positive values are returned for the region that captures most of the data in the target class (in-class), and negative values outside this region (out-of-class).

Assuming a set of N n-dimensional training vectors $\mathbf{x}_i \in R^n, i = 1, \ldots, N$, that do not contain any class information, the ν-SVM can be solved by the following quadratic optimization problem [Chang and Lin 2011; Schölkopf et al. 2001; Yang et al. 2010]:

$$\begin{aligned} \min_{\boldsymbol{\omega}, \boldsymbol{\xi}, \rho} \quad & \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & \boldsymbol{\omega}^T\phi(\mathbf{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, i = 1, \ldots, N, \end{aligned} \tag{12}$$

where $\phi(\mathbf{x}_i)$ is a kernel function that maps the $\mathbf{x}_i$ into a higher dimensional feature space. The parameter $\nu \in (0, 1]$ was introduced by Schölkopf et al. [2000] to provide an effective control on the lower bound of support vectors (as the fraction of training records). Schölkopf et al. [2000] also proved that ν provides an upper bound on the training error. For the one-class SVM, ν provides an upper bound on the fraction of training vectors that are classified as being out-of-class. The decision

function resulting from Equation 12 is [Chang and Lin 2011; Schölkopf et al. 2001]:

$$d(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho\right), \tag{13}$$

where $\boldsymbol{\alpha}$ is the solution to the dual problem of Equation 12, and $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel matrix. Vectors $\mathbf{x}_i$ with $d(\mathbf{x}_i) \geq 0$ will be classified as in-class (normal) while those with $d(\mathbf{x}_i) < 0$ are classified as out-of-class (abnormal).

For our **SVM** classifier, we train a ν-SVM using as training set the feature vectors $\mathbf{f}_i = features(v_i)$ generated from all $v_i \in T_t$, i.e. only values that occur frequently in V, where $features(v_i)$ is a function that calculates and returns the numerical features from Table II for a given input value v_i. The ν-SVM therefore learns the feature characteristics of frequent values. As we did for the other approaches, we also calculate the training normal percentage $trnp$ on the subset $T_s \subset T$ by calculating how many values in T_s are within the decision boundary:

$$trnp_{svm}(T_s) = |\{v_j \in T_s : d(\mathbf{f}_j) \geq 0\}|/|T_s|, \tag{14}$$

where $\mathbf{f}_j = features(v_j)$. Next we apply the trained ν-SVM on all values in S. For each feature vector $\mathbf{f}_j$ generated from $v_j \in S$, the classifier returns the distance $d_j = d(\mathbf{f}_j)$ of $\mathbf{f}_j$ from the SVM decision boundary. We use this distance as our abnormality score:

$$a_{SVM}(v_j) = d_j \tag{15}$$

for the **SVM** classifier, and rank all $v_j \in S$ according to their d_j values. A zero or positive value for d_j means an attribute value is classified as normal, while a negative value for d_j means it is classified as abnormal. Attribute values with high negative distances d_j have feature vectors that differ most from the feature vectors of normal values, and these are the most abnormal values.

The computational complexity of learning the ν-SVM is quadratic in the number of training records, i.e. $|T_t|$, and the number of iterations used by the algorithm [Chang and Lin 2011; Schölkopf et al. 2001], while the evaluation of a single test record depends linearly upon the number of support vectors which can be controlled through the parameter ν [Schölkopf et al. 2000].

6.1. Feature Ranking

Depending upon the domain and characteristics of the data set analyzed, we expect different features to be relevant for distinguishing normal from abnormal values. Knowing which features are most relevant will allow a set of rules to be generated based on these features. Such rules can then be used to analyze attribute values according to these features, and therefore facilitate efficient automatic detection of abnormal values within operational data cleaning systems.

To investigate which of the features from Table II are most useful in distinguishing normal from abnormal values, we conduct feature ranking using a supervised classification approach. We first generate two training sets, one each for the 'normal' and 'abnormal' classes based on the values $v_i \in V$ as classified by the trained ν-SVM. Using the learned decision function d from Equation 13, we calculate the distance d_i for all $v_i \in V$ based on their feature vectors $\mathbf{f}_i$: $d_i = d(features(v_i))$, and then add only the fraction $r \times |V|$ (with $0 < r < 1$) of feature vectors with the largest positive and largest negative distances into the training sets F_N of normal ($d_i \geq 0$) and F_A of abnormal ($d_i < 0$) feature vectors, respectively.

Based on the F_N and F_A training sets, we next employ a decision tree classifier which calculates information gain [Quinlan 1986] (based on entropy) for each feature, where an optimal numerical threshold is calculated based on the normalized feature values. A higher information gain means a feature is better able to distinguish between normal and abnormal attribute values.

We next report on implementation details of our different techniques and the experimental methodology we employ, and we describe the data sets we used in our evaluation.

7. EXPERIMENTS

We implemented our techniques in Python (version 2.7.3) using the Scikit-learn[1] machine learning package [Pedregosa et al. 2011] (which is based on the LIBSVM [Chang and Lin 2011] library) for the **SVM** classifier, while the **BSI** and **PLM** techniques are pure Python implementations. The program codes are available from the authors.

7.1. Parameter Settings

We used various parameter settings to investigate the behavior of our proposed techniques. For the main parameters required in Algorithm 1, we set the minimum count m for an attribute value to be included into the training set T as $m = [2, 3, 4, 5, 7, 10, 20, 50]$, while all values that occur less than m times were added into the set S of potentially abnormal values. To calculate the training normal percentages, $trnp$, we set $t = 0.9$ to include 90% of all values in T into T_t and 10% into T_s.

For the **BSI** and **PLM** classifiers we report results for $q = [1, 2, 3]$, and for **PLM** we set the model's order to $n = [1, 2, 3]$. We trained the **SVM** with linear and non-linear (polynomial and RBF) kernels. We set the **SVM** parameter ν [Schölkopf et al. 2001] to $\nu = [0.01, 0.02, 0.05, 0.1]$. We also present the times required for training and testing (classification) of our different techniques. For feature ranking of the **SVM** classifier we set $r = 0.5$, which means we selected the 50% of feature vectors that are furthest away from the decision boundary into the training sets F_N (assumed normal) and F_A (assumed abnormal). We used the decision tree classifier available in the Scikit-learn [Pedregosa et al. 2011] package to calculate the feature ranking based on information gain.

We do not provide a comparative evaluation of our proposed techniques with other abnormality detection techniques for categorical data, such as those presented in Section 2. Based on our analysis of the discussed related works, we did not find any approach that we believe allows a meaningful comparison for three main reasons. The first is that a technique is aimed at detecting abnormal values in several attributes (all techniques discussed in Section 2.1), i.e. a technique uses the characteristics of one attribute to identify abnormal values in another attribute. The second is that a technique was developed for different types of data compared to ours, such as long biological sequences, and aims to detect abnormal sub-sequences (the techniques discussed in Section 2.2). The third is that a technique is aimed at clustering attribute values and assumes that rare values are variations of a given frequently occurring value (as described in Section 2.4). Additionally, techniques in this latter category have high computational complexities (quadratic in the case of Ciszak [2008] and log-linear for the approach by Kazimianec and Augsten [2011], both in the number of attribute values) and have only been applied to relatively small domains of less than 40,000 values.

7.2. Data Sets

We evaluate our approaches with four large real-world data sets. All are publicly and freely available. The first two are versions of the North Carolina Voter Registration database[2] accessed on the 5th of April 2014 (named 'NCVR-A') and 3rd of October 2014 (named 'NCVR-O'), respectively. These data sets contain the names, addresses, phone numbers, and other personal details, of over six million voters [Christen 2014]. During processing of the NCVR-A data set we identified a significant number of records that contain corrupted values in some attributes, which partially motivated this research. We do not know the reason why this database was badly corrupted, but we suspect errors in an automated data entry process (such as during optical character recognition) that were not detected and corrected. Applying our techniques during data entry and processing would have helped to automatically discover such corrupted data.

The third data set is the 2013 KDD Cup data (named 'KDD') which contains bibliographic author and publication records from Microsoft Academic Search as available from *Kaggle*[3]. While the topics of the 2013 KDD Cup were about author-paper identification and author disambiguation, in this

[1] http://scikit-learn.org
[2] ftp://alt.ncsbe.gov/data/
[3] https://www.kaggle.com/c/kdd-cup-2013-author-paper-identification-challenge/

Table III: Data set characteristics, where c_i is the number of times a value occurs in an attribute.

Attribute	Unique values	Max. c_i	Perc. $c_i = 1$	Perc. $c_i = 2$	Perc. $c_i = 3$	Perc. $c_i > 3$
NCVR-A (April 2014) – 6,293,509 records						
First name	202,395	207,485	19.0 %	47.9 %	4.8 %	28.4 %
Last name	283,869	122,396	12.7 %	36.2 %	3.2 %	47.9 %
Street address	3,108,161	4,414	30.8 %	36.9 %	11.6 %	20.7 %
Phone number	2,015,348	14,296	51.6 %	37.3 %	2.5 %	8.6 %
NCVR-O (October 2014) – 7,539,858 records						
First name	232,783	141,384	65.2 %	13.3 %	5.3 %	16.2 %
Last name	331,645	95,021	45.5 %	16.5 %	8.1 %	29.9 %
Street address	3,591,547	4,550	36.6 %	36.8 %	15.8 %	10.8 %
Phone number	2,350,104	13,367	78.1 %	17.1 %	3.4 %	1.4 %
KDD – 12,776,670 records						
Author name	2,456,026	5,105	58.3 %	15.7 %	7.0 %	19.0 %
Paper title	1,981,069	670	95.1 %	4.5 %	0.3 %	0.1 %
Author affiliation	830,288	311,386	48.3 %	20.6 %	10.9 %	25.5 %
SNAP – 41,036,252 records						
Phrases	14,542,030	29,402	68.5 %	15.3 %	5.3 %	10.9 %

work we are interested in discovering abnormal authors names, paper titles, and author affiliation (institution) names. While this data set contains records from many different countries, some written in different languages and character encodings, we limited our analysis to values made of only ASCII characters and removed all values that contain non-ASCII characters. Note however that our techniques work on any Unicode character encoding.

The final data set is the *Memetracker* data set[4] (named 'SNAP') from the *Stanford Network Analysis Project* (SNAP) [Leskovec and Krevl 2014] which contains short phrases that appeared frequently over time across online news and social networking sites. We used the April 2009 subset which contains over 41 millions mostly short text values.

Some basic characteristics of the four data sets are summarized in Table III. The minimum length of a value to be included into the set V (and on which the counts given in this table are based on) is two characters. As can be seen, the unique number of attribute values is over 200,000 for all attributes, and reaches over 14 millions for the SNAP data set. The percentage of values that occur only once ($c_i = 1$) or a few times ($c_i = 2$ or $c_i = 3$) is substantial and ranges from around 50% to over 99%, while for all attributes some values occur very frequently.

Note that we do not have ground truth data (known normal and abnormal values) in any of these data sets. Given this lack of ground truth, one way to evaluate the accuracy of our techniques would be to generate synthetic data consisting of normal and abnormal values. However, we do not believe there is a meaningful way to generate realistic abnormal values that would reflect the characteristics of real abnormal values that could be expected in practical applications. As normal values we can extract values that occur frequently in a data set. Any data generation would require functions and parameter settings that determine how specific abnormal values are created (such as how to modify values or where to insert certain random characters, and how many), and this would necessarily result in a biased detection of those created abnormal values. It would be impossible to generate all types of realistic variations, errors, and modifications that might occur in real data sets.

To evaluate the comparative accuracy of our three approaches, we have instead conducted an experiment with a manual evaluation of randomly selected normal and abnormal values that were manually classified as normal or abnormal by three of the co-authors. We report on this experiment in Section 8.3.

[4] http://www.memetracker.org/

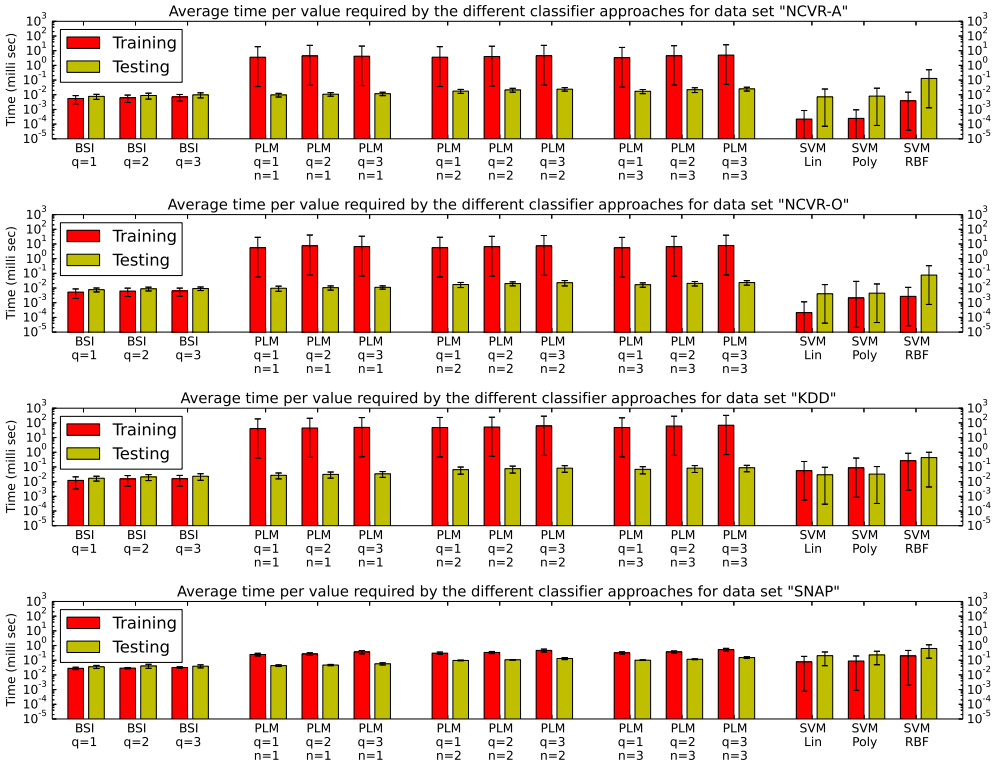


Fig. 2: Timing results for the four data sets for the different classifiers (averages and standard deviations over all parameter settings – except n and q – per classifier), calculated as the average time required per value. Note the y-axis is in logarithmic scale.

8. RESULTS AND DISCUSSION

We now present and discuss experimental results obtained on the three techniques and four data sets described above. We describe timing, and training normal and testing abnormal percentage results, a manual classification of selected values, explore the distributions of normal and abnormal value scores, evaluate feature rankings, show selected top ranked normal and abnormal example values, and finally provide an overall discussion of our experimental evaluation.

8.1. Training and Testing Times

We start our discussion of results by looking at the timing performance of training and testing of our three classifiers, as shown in Fig. 2. As can be seen, and as expected, for the NCVR and KDD data sets the simple **BSI** technique is faster than the more sophisticated **PLM** and **SVM** classifiers. The **PLM** technique is the slowest with regard to training time, but requires time comparable to the **BSI** technique to classify a value as being normal or abnormal. Neither the length q of q-grams nor the order of the model n had any significant influence on the training and testing times required.

The **SVM** classifier is relatively faster, which is due to it being based on the efficient LIBSVM [Chang and Lin 2011] library. We would expect the **BSI** and **PLM** techniques to be significantly faster as well if implemented in Java or C. Still, the results for the **BSI** and **PLM** techniques show their relative timing requirements with regard to different parameter settings. Interestingly, the times for all classifiers for the SNAP data set are very similar. This is likely because the average

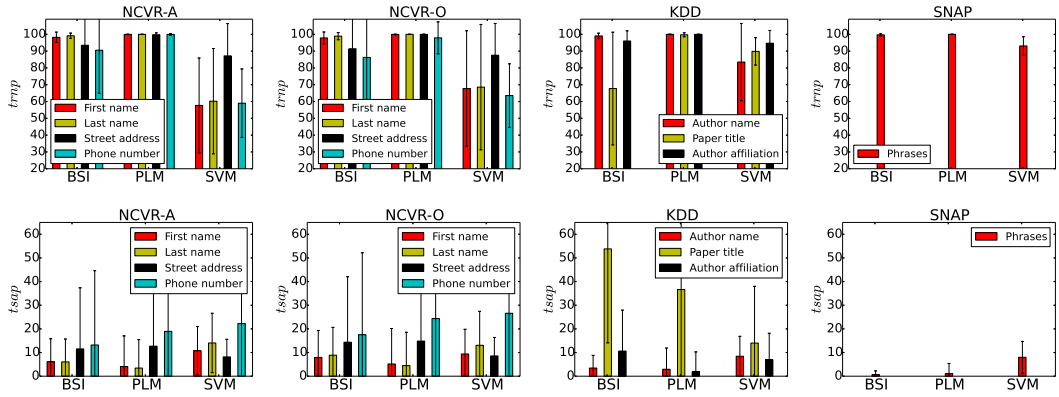


Fig. 3: Training normal percentages ($trnp$) in top row and testing abnormal percentages ($tsap$) in bottom row for all attributes averaged over different classifier parameter settings on the four data sets.

length of values is much longer in the phrases in this data set compared to the shorter values in the other data sets, making the testing phase slower in comparison to the other data sets.

As can also be seen from Fig. 2, the testing times for all techniques are well below one millisecond for a single value, making all three techniques candidates for real-time discovery of abnormal values, and therefore suitable for implementation in practical data cleaning applications where all attribute values in streams of records need to be identified if they are abnormal before being processed further or added to a database.

8.2. Training Normal and Testing Abnormal Percentages

The training normal percentages, $trnp$, and testing abnormal percentages, $tsap$, for the different data sets and their attributes are shown in Fig. 3. We emphasize these measures are not aimed at evaluating the accuracy of our approaches (because we do not have truth data), rather they allow us to compare the characteristics of our approaches for detecting abnormal values. From this figure we can see that the **BSI** and **PLM** techniques lead to high to very high $trnp$ results, which means they are accurate in the sense that they classify the vast majority of normal values as being normal. An exception is **BSI** on the 'Paper title' attribute of the KDD data set, which highlights that there are some unusual q-grams in paper titles that occur quite frequently. Note that for this attribute the training data set can become very small, as for $c \geq 3$ only 0.3% of the nearly 2 million unique paper titles (i.e. around $6,000$ values) can be used for training.

The **SVM** classifier on the other hand, at least for the two NCVR data sets, has much lower $trnp$ results. This indicates that there are a significant number of attribute values in these data sets that have unusual features and occur quite frequently. Looking at the testing abnormal percentages, $tsap$, in the bottom row of Fig. 3, it is clearly visible that some attributes have more abnormal values than others as all three classifiers show higher $tsap$ values. Exceptionally high $tsap$ values are shown for the 'Paper title' attribute of the KDD data set (especially with the **BSI** classifier), which correlates with the low $trnp$ values for the **BSI** classifier on this attribute, as well as the 'Phone number' attribute of the NCVR data sets.

Drilling down into the three **SVM** kernel and the three **PLM** smoothing methods shown in Table IV, we can see that the linear and polynomial kernels exhibit very similar behavior, while the RBF kernel for most data sets has significantly lower $trnp$ values, and higher $tsap$ values for all data sets. All **PLM** smoothing methods show very high $trnp$ values, but absolute discount smoothing leads to higher $tsap$ values compared to the other two smoothing techniques, with Good-Turing having the overall smallest $tsap$ values. This means that the **SVM** classifier with RBF kernel and

Table IV: Training normal percentages $trnp$ and testing abnormal percentages $tsap$ (averages and standard deviations) for different **SVM** kernels and **PLM** smoothing methods, averaged over attributes.

	NCVR-A	NCVR-O	KDD	SNAP	NCVR-A	NCVR-O	KDD	SNAP
	Training normal percentages $trnp$				Testing abnormal percentages $tsap$			
SVM Lin	83.4±13.2	89.2±10.7	95.4±5.6	94.1±4.4	10.0±14.8	9.4±15.4	2.7±3.0	6.2±5.0
SVM Poly	83.6±12.7	88.7±12.0	95.5±5.5	94.1±4.4	10.7±15.6	10.5±16.4	2.7±3.2	6.2±5.0
SVM RBF	47.5±29.8	49.8±29.4	77.1±20.7	91.0±6.7	21.7±15.5	21.8±17.0	23.9±21.9	11.5±8.1
PLM AbsDisc	99.9±0.7	99.6±4.2	99.9±0.7	100±0.0	15.5±29.8	19.8±33.2	21.1±35.6	1.5±5.3
PLM Laplace	99.9±0.5	99.2±5.9	99.9±0.6	100±0.0	8.1±23.5	10.1±25.7	11.6±28.9	0.5±1.5
PLM GoodTu	99.9±0.7	99.6±4.2	99.9±0.7	100±0.0	5.8±18.2	6.7±19.1	8.8±24.1	0.5±1.6

the **PLM** classifier with absolute discount smoothing have reported more rare values to be abnormal compared to the other parameter settings for these two approaches.

In Fig. 4 we show the $trnp$ for different minimum training value frequencies, m, for the different data sets as averaged over attributes and parameter settings. For the **BSI** classifier, as one would expect, with larger values of q (i.e. longer q-grams) and higher values of m the value of $trnp$ decreases as less q-grams are included in the training data. This leads to higher percentages of values classified as abnormal, especially for $q = 3$. For the SNAP data set, which is much larger than the other three data sets, $trnp$ stays high for all values of m which means most q-gram tuples seem to occur more than 50 times. This is to be expected given the vast majority of the SNAP attribute values are normal text phrases.

For the **PLM** classifier, $trnp$ stays high for all values of q except with the largest value $m = 50$ for the NCVR-O (October 2014) data set where it drops slightly. This indicates there is a difference in the characteristics of values that occur less than 50 times compared to those that occur more than that. Both the linear and polynomial kernels of the **SVM** classifier achieve consistently high (above 80%) $trnp$ values, however for the RBF kernel the results on both NCVR data sets are very low for small values of m. As can be seen in Figure 4, for low values of m (low threshold for values to be included into the training set S), low $trnp$ results are obtained for the RBF kernel when compared to the linear and polynomial kernels. It seems that with the RBF kernel the one-class SVM classifier is generating a decision boundary that is overfitting the training data and therefore more values in S are classified as being abnormal. For low values of m there will be a much larger diversity of attribute values (including some truly abnormal values) in S leading to a higher diversity of feature vectors which allow a more accurate decision boundary to be learned compared to when m becomes larger (leading to a less diverse training set and corresponding less diverse set of feature vectors). From Figures 4 and 5 we can see that with the RBF kernel **SVM** approach both the $trnp$ and $tsap$ increase with larger m, and therefore overfitting becomes less with larger values of m.

The testing abnormal percentages, $tsap$, shown in Fig. 5, indicate that for all classifiers and all parameter settings with larger values of m more attribute values (that occur less than m times) are classified as abnormal. This is expected, because with larger values for m the training set T_t becomes smaller, and therefore less unique q-grams are included in T_t for the **BSI** and **PLM** classifier, while for the **SVM** classifier the variety of feature vectors also becomes smaller. Overall, the increase in $tsap$ is consistent across classifiers and their parameter settings, with the increases for the individual data sets being similar across classifiers. For values of $m \leq 5$, $tsap$ is generally below 10% (except for the KDD data set and **BSI** classifier), however the **SVM** classifier with the RBF kernel overall has a much higher $tsap$ rate, which means it will classify more rare values to be abnormal.

8.3. Comparative Manual Classification

To provide quantitative measures on how well our three techniques perform in detecting abnormal values, we conducted an experiment where three of the co-authors manually classified sets of selected attribute values as being normal or abnormal. For each attribute in the four data sets and each of the three techniques, we randomly selected 100 normal and up-to 100 abnormal attribute values (for some attributes only a few values were automatically classified as abnormal by our techniques).

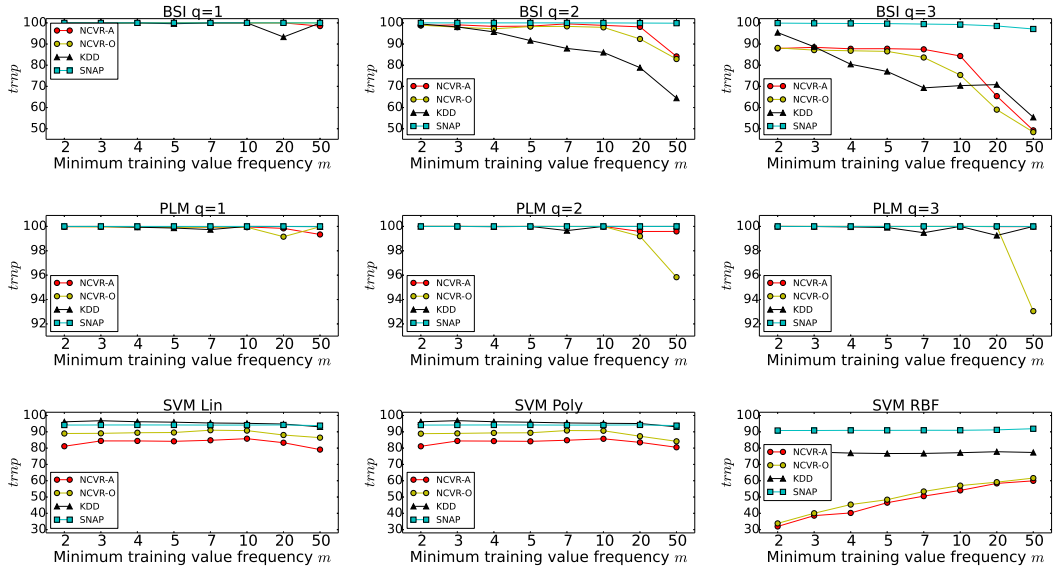


Fig. 4: Training normal percentages ($trnp$) for increasing minimum training value frequencies (m) included into the training set T for the three classifier approaches (one per row), with selected parameter variations and averaged over attributes for each data set. Note that each row has a different y-axis scaling.

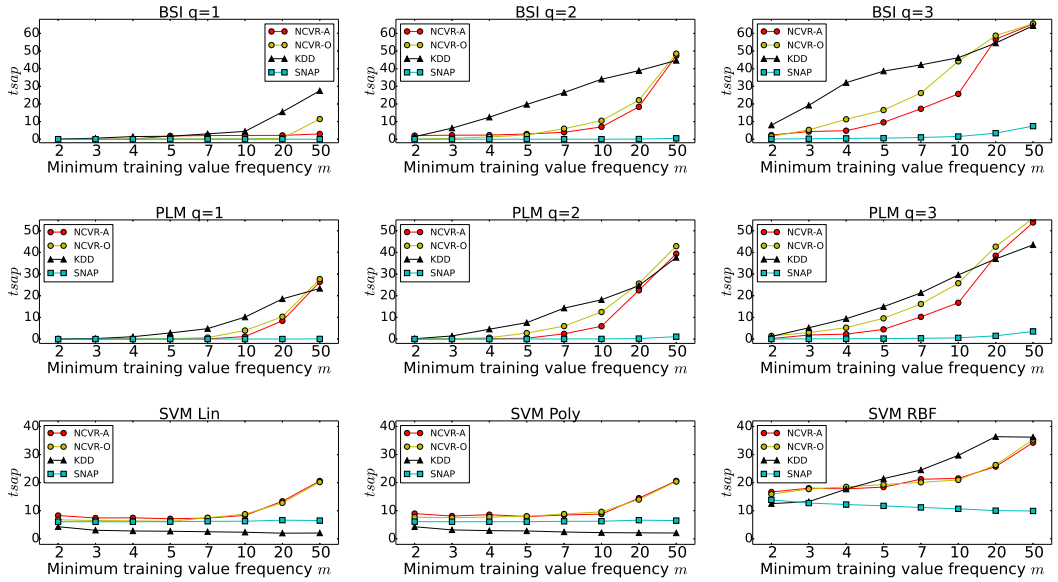


Fig. 5: Testing abnormal percentages ($tsap$) for increasing minimum training value frequencies (m) included into the training set T for the three classifier approaches (one per row), with the same parameter variations as in Fig. 4 and averaged over attributes for each data set. Note that each row has a different y-axis scaling.

Table V: Comparative accuracy results (as precision and recall) of automatic and manual classification of abnormal values of a sub-set of randomly selected attribute values for the three classifiers. Results for attributes with both precision and recall as 0.0 are where no attribute value was manually classified as abnormal.

	BSI		**PLM**		**SVM**	
	Precision	Recall	Precision	Recall	Precision	Recall
NCVR-A (April 2014)						
First name	0.11	1.0	1.0	1.0	0.05	1.0
Last name	0.26	1.0	1.0	1.0	0.03	1.0
Street address	0.01	1.0	0.0	0.0	0.04	1.0
Phone number	1.0	1.0	1.0	1.0	0.02	1.0
NCVR-O (October 2014)						
First name	0.0	0.0	1.0	1.0	0.08	1.0
Last name	0.0	0.0	0.67	1.0	0.02	1.0
Street address	0.01	1.0	0.67	1.0	0.03	1.0
Phone number	0.25	1.0	0.0	0.0	0.01	1.0
KDD						
Author name	0.13	0.93	0.0	0.0	0.0	0.0
Paper title	0.01	1.0	0.01	1.0	0.06	0.86
Author affiliation	0.04	0.80	0.0	0.0	0.37	0.97
SNAP						
Phrases	1.0	1.0	1.0	0.50	0.25	0.96

These were then manually assessed as being normal or abnormal. For each value, the majority class of the three manual classifications was used as the 'true' manual label that was compared to the automatic label as determined by one of our three techniques. Table V shows the obtained comparative accuracy results as precision and recall values [Christen 2012] (between 0.0 and 1.0) for the class of abnormal values. A true positive was a value classified as abnormal by both one of our techniques and at least two of the three manual assessments. A false positive was a value manually classified as normal but classified as abnormal by a technique, while a false negative was a value manually classified as abnormal but normal by a technique. For some attributes no values were manually classified as abnormal, and therefore both precision and recall measures became 0.0

Ideally, this manual assessment should lead to high recall but possibly low precision values, i.e. all manually classified abnormal values should also be automatically classified as abnormal by our techniques (i.e. our techniques should not miss abnormal values), while our techniques should classify more values as abnormal than the manual assessment in order to provide users in an operational setting with a large variety of potentially abnormal values. As can be seen from Table V, for most data sets and attributes we do obtain very high to perfect recall results. The low recall for the SNAP 'Phrases' attribute is due to the multi-lingual content of this data set where the manual assessment led to some non-English phrases being classified as abnormal even though these were normal values according to our three techniques.

8.4. Distributions of Abnormality Scores

Figures 6a to 6c show histograms with the distributions of abnormality scores. For a given classifier and attribute, for each attribute value in S we averaged its abnormality score (following one of Equations 2, 10, and 15) over all parameter settings, and the final 'normal' or 'abnormal' classification was based on the majority classification for an attribute value (i.e. if for more than 50% of parameter settings an attribute value was classified as 'abnormal' then its final classification was also 'abnormal'). Therefore, these histograms show how different classifiers score rare attribute values in S with regard to their abnormality.

As can be seen, the **BSI** and **SVM** approaches classify a much larger number of values as abnormal compared to the **PLM** approach. In two occasions, the final classification for almost all values with the **PLM** approach was normal (NCVR-A 'Street address' and KDD 'Author affiliation'). As the bar plots in Figure 3 show, the $tsap$ values for most attributes have a high variance (high standard deviation as shown in the vertical black lines) over the different parameter settings of

the three classifiers. The variances are especially high for the NCVR-A 'Street address' and 'Phone number' attributes for the **BSI** and **PLM** classifiers. As a result, for different parameter settings different attribute values are classified as abnormal or normal. In Figure 6a, as a result of the majority classification of each attribute value over the different parameter settings, and the large variance of classifications (as visible in Figure 3), it seems the vast majority of individual attribute values for the NCVR-A 'Street address' and 'Phone number' attributes for the **BSI** and **PLM** classifiers are more often classified as normal than as abnormal, and only few are more often classified as abnormal than normal.

The three classifiers also exhibit very different distributions of abnormality scores. Both the **BSI** and **SVM** approaches show clearly skewed distributions towards normal values, while the **PLM** approach shows a somewhat more uniform, however rugged, distribution of scores. In future work we aim to investigate such scores further to see if they can be directly useful to assess the quality of values in a given attribute, and potentially can help to conduct an overall ranking of attributes with regard to their 'abnormality'.

8.5. Feature Rankings

This brings us to the feature ranking results given in Table VI which shows the top features (with information gain (IG) values of 0.1 or larger) for the different data sets and kernels for the **SVM** classifier. Note that in many cases the linear and polynomial kernels led to very similar feature ranking results including highly similar IG values, which indicates their performances are not statistically different. As can be seen from this table, only a small subset of features is being used by the **SVM** classifier, and many of these are used across data sets and attributes. The most commonly used features are the different length-based ones (value or word length). For the SNAP data set, length- and repetition-based features are also ranked highly.

For the phone number attribute in the NCVR data sets (the only attribute where we would expect most normal values to have only digits) the number of digits and repetition-based features are used to distinguish between normal and abnormal phone numbers. We note that broadly speaking there are three types of abnormal phone numbers: those containing the wrong number of digits, those containing letters or other inappropriate characters, and those that are essentially normal phone numbers but might be fake, such as 123 456 7890. Our experiments indicate that the **SVM** approach is effective at detecting the first type, while the **BSI** and **PLM** approaches are effective at detecting the second type. However, detecting the third type is problematic since some of these will in fact be normal phone numbers. After all, someone (possibly) has the phone number 123 456 7890.

8.6. Selected Abnormal and Normal Example Values

To allow the reader to appreciate what actual discovered abnormal values look like, in Tables VII to X we show selected abnormal as well as normal attribute values that had either high abnormal or high normal scores, respectively, as averaged over different parameter settings for a given data set, attribute, and classifier. Some discovered abnormal values were too long to be fully included in these tables and we therefore had to truncate them.

As can be seen, all three approaches are capable of discovering abnormal values, with some values ranked highly by several approaches. While some exceptionally abnormal values were scored highly by more than one of our classifiers, each of the three approaches (**BSI**, **PLM** and **SVM**) ranked values with different characteristics as being highly abnormal.

Overall, it seems the q-gram based classifiers, **BSI** and **PLM**, are better suited to discover abnormal values than the **SVM** classifier, as can be seen for example in the abnormal values from the NCVR-O data set in Table VIII. The **PLM** is for example able to identify unusual phone numbers with regard to a certain number range, such as unusual area codes in the NCVR data sets that have a rare combination of numbers (like 401 or 601), as shown in Tables VIII. The **SVM** classifier seems to favor shorter values, which is in line with the top-ranked features used by this approach as can be seen from Table VI.

Table VI: Top **SVM** features for the four data sets, ranked according to their information gain (averages and standard deviations are shown as calculated over all parameter settings). Only features with average information gain of at least 0.1 are included (thus the shown values do not necessarily sum to 1.0).

Attribute	Linear kernel		Poly kernel		RBF kernel	
NCVR-A (April 2014)						
First name	**AvrWLen**	0.50±0.46	**MinWLen**	0.47±0.47	**MinWLen**	0.43±0.23
	RatLetLen	0.22±0.35	**AvrWLen**	0.25±0.42	**ValLen**	0.35±0.20
	MinWLen	0.21±0.39	**RatLetLen**	0.24±0.35		
Last name	**AvrWLen**	0.47±0.46	**AvrWLen**	0.47±0.46	**ValLen**	0.49±0.11
	MinWLen	0.44±0.44	**MinWLen**	0.45±0.45	**MinWLen**	0.21±0.21
					AvrWLen	0.10±0.17
Street address	**ValLen**	0.85±0.27	**ValLen**	0.88±0.23	**RatLetLen**	0.29±0.27
					ValLen	0.17±0.21
Phone number	**NumDigit**	0.56±0.47	**NumDigit**	0.55±0.48	**LonRepChr**	0.52±0.33
	LonRepChr	0.34±0.39	**LonRepChr**	0.34±0.39	**LonRepSeq**	0.18±0.33
					LonIncSeq	0.12±0.11
					LonDecSeq	0.11±0.14
NCVR-O (October 2014)						
First name	**NumLow**	0.55±0.49	**NumLet**	0.49±0.46	**MinWLen**	0.47±0.25
	NumLet	0.43±0.49	**NumLow**	0.45±0.48	**ValLen**	0.40±0.20
Last name	**NumLow**	0.40±0.42	**NumLow**	0.38±0.43	**ValLen**	0.53±0.14
	NumLet	0.24±0.38	**NumLet**	0.36±0.40	**MinWLen**	0.24±0.22
	MaxWLen	0.22±0.38	**RatPunctLen**	0.14±0.18		
Street address	**ValLen**	0.64±0.42	**ValLen**	0.75±0.36	**RatLetLen**	0.35±0.25
	NumLow	0.18±0.33	**NumLow**	0.11±0.28	**ValLen**	0.20±0.18
	NumLet	0.12±0.29				
Phone number	**NumDigit**	0.43±0.43	**LonRepChr**	0.51±0.33	**LonRepChr**	0.40±0.33
	LonRepChr	0.24±0.34	**NumDigit**	0.25±0.27	**LonRepSeq**	0.28±0.35
	ValLen	0.16±0.36				
KDD						
Author name	**NumLet**	0.75±0.18	**NumLet**	0.73±0.22	**RatPunctLen**	0.20±0.20
					AvrWLen	0.16±0.13
					NumUppW	0.13±0.12
Paper title	**NumLet**	0.67±0.29	**NumLet**	0.68±0.29	**ValLen**	0.80±0.21
	RatLetLen	0.30±0.30	**RatLetLen**	0.29±0.30	**NumLow**	0.18±0.20
Author affiliation	**NumUpp**	0.61±0.23	**NumUpp**	0.61±0.23	**ValLen**	0.74±0.23
	RatPunctLen	0.14±0.19	**RatPunctLen**	0.14±0.19		
	NumLet	0.12±0.09	**NumLet**	0.12±0.09		
	RatLetLen	0.11±0.08	**RatLetLen**	0.11±0.08		
SNAP						
Phrases	**RatLetLen**	0.56±0.36	**RatLetLen**	0.56±0.36	**ValLen**	0.53±0.06
	LonDecSeq	0.20±0.33	**LonDecSeq**	0.20±0.33	**NumAlphaW**	0.35±0.02
	AvrWLen	0.19±0.32	**AvrWLen**	0.19±0.32		

However, all three classifiers are able to identify abnormal values and provide a user with a picture on the quality of attribute values in their data sets. For example, from Tables VII and VIII it is obvious that three of the four attributes (first and last name, and phone number) in the NCVR-A (April 2014) data set suffer severe data quality problems, with completely corrupted values, and values that seem to have spilled across attributes or that were stored in the wrong attribute. On the other hand, the highly scoring abnormal values in the NCVR-O (October 2014) data set (Table VIII) show only minor data quality issues, which are likely optical character recognition (OCR) problems, such as '0' (zero) instead of 'o', or a bracket ')' instead of the '1' digit.

Similar data quality problems occur in the KDD data set as can be seen from Table IX. Some abnormal values seem to occur in the wrong attribute, while others are clearly made-up, meaningless values (such as ABCDE; FGHIJ; KLMNO). The identified abnormal phrases in the SNAP Meme-

tracker data set shown in Table X highlight that some values in this data set are not meaningful comments but rather some kind of codes, or Unicode text that was not properly encoded.

8.7. Discussion

As the abnormal examples in Tables VII to X have shown, our approaches can provide a user with an immediate view on the quality of their data in an efficient and automatic way. If the top scored abnormal values look correct or only include minor corruptions (such as OCR errors), or are unusual but valid attribute values (such as rare surnames from a different culture, or rare rural addresses) that otherwise follow what is expected in an attribute, then this confirms that the quality of the values in the attribute is high. On the other hand, if the top scored values look completely different from what is expected (such as in the case of the NCVR-A data set shown in Table VII), then this immediately highlights significant data quality problems that need to be corrected.

Of the three classifiers, the **BSI** approach is sensitive to rare characters which lead to q-grams that may only occur in a single value (for example an OCR error where a letter is replaced by a digit). The **PLM** approach is more robust and, unlike the **BSI** classifier, can also score values as abnormal where all q-grams have occurred in the training set. Finally, the **SVM** classifier depends upon the features (as shown in Table II) used. For different data sets and attributes it will be of advantage to customize these features depending upon the expected content of these attributes.

The primary assumption of our approach is that there are no frequent values that are abnormal. We can imagine that there are however certain situations where abnormal values may occur with sufficiently high frequency as to be included in the set of normal values T. Some possible examples are systematic OCR problems, or where a single person is responsible for testing and repeatedly entering the same 'incorrect' values over and over. However, in most cases this will result in only a few frequent abnormal values (with counts $> m$) being missed while other abnormal values (counts $< m$) are still being detected.

As an example, if an OCR system incorrectly classifies 5% of instances of the digit 5 as the letter S in a postcode or zipcode attribute, then while a few values may be common enough that they are included in the set of normal values, the probability attached to q-grams that include the letter S will still be significantly lower than all the correct digit-only q-grams (assuming a relatively even distribution of digits). As a result, when using the **PLM** approach we would still assign high abnormality scores to values which contain q-grams that include the letter S and they may well show up as abnormal values. We note that if the error is truly widespread, for example an OCR system that gets it wrong more often than it gets it right, then it is possible that the learned model has probabilities that make it impossible for our techniques to detect this type of error. However, we like to think this type of situation is very unusual in practice. We also note that even in an extreme case such as this, our techniques could still detect abnormal values with different characteristics..

In practical data cleaning applications, our approach can be employed in two different scenarios. The first would be the batch cleaning of a given data set, where one or several of the presented classifiers is (are) trained and applied on the data set, and rare values are ranked according to their abnormality scores. Based on operational requirements (such as time and resource limitations), a certain number of top ranked abnormal values can then be manually assessed and validated as being true abnormal and unexpected values. This set of validated abnormal values can then for example be used to learn data cleaning or extraction rules. In this scenario, the abnormality scores are used to select the most abnormal values for manual assessment rather than the classification threshold we used in our experiments. In the second scenario, attribute values from a stream of records are classified in real-time, and those with an abnormality score above a given threshold are either removed or flagged for manual assessment. In this scenario, setting a suitable threshold will be crucial given likely operational requirements and limitations.

The output of our approaches, normal and abnormal values and their abnormality scores, can be directly used as training data for supervised outlier detection or data extraction techniques that either build a more refined model of abnormal values, or that generate rules that allow similar abnormal

values to be detected in the future at the time they are entered by a user or crawled from a Web site, thereby preventing abnormal values to be included in a database in the first place.

9. CONCLUSIONS AND FUTURE WORK

We have presented three approaches for automatically discovering abnormal (unusual and unexpected) values in large textual databases. One is a simple set-based approach that uses character q-grams extracted from attribute values, the second approach extends this q-gram based approach by incorporating probabilities calculated using a probabilistic language model, and the third approach uses morphological word features to train a one-class SVM classifier. Experiments on four large real-world data sets have shown that these approaches can successfully discover abnormal textual values, as illustrated by the examples shown in Tables VII to X.

For future work, we plan to investigate how the results of the different approaches can be fused together by using ensemble methods to improve the final results presented to a user. We will further investigate the features used by the one-class SVM classifier, and try to identify novel features that might be better suited for the task of discovering abnormal textual values. We also plan to investigate how to best select parameter values for our approaches, such as the q-gram length q for the **BSI** and **PLM** classifiers, and the order n of the **PLM** model. Using different values for q and n depending upon the length of an attribute value can potentially improve the detection performance of our approaches.

While we currently only consider attributes independently, we also aim to extend our approaches to identify values that are abnormal in their current attribute but normal in another attribute, indicating such values are stored in the wrong attribute and should potentially be stored in a different attribute.. Our approaches can also be used in a 'boot-strapping' process where first a set of abnormal values is discovered, then assessed manually, and then the sets of normal and assessed abnormal values are used to train a fully supervised classifier. The set of abnormal values can also be expanded by including known abnormal values not identified with our approaches.

The proposed approaches have limitations in that they are not able to discover abnormal values that have characteristics similar to normal values. For example, our approaches would unlikely classify Donald Duck or Simple Test as an abnormal name. We aim to investigate other techniques that would allow us to identify such values. Our ultimate aim is to develop techniques that allow for automatic real-time identification of abnormal values during data entry, and therefore prevent storage of bogus, inappropriate, or wrong values in a database in the first place.

REFERENCES

Maik Anderka, Benno Stein, and Nedim Lipka. 2011. Detection of Text Quality Flaws as a One-class Classification Problem. In *CIKM*. ACM, Glasgow, 2313–2316.

Alberto Apostolico, Mary Ellen Bock, Stefano Lonardi, and Xuyan Xu. 2000. Efficient detection of unusual words. *Journal of Computational Biology* 7, 1–2 (2000), 71–94.

Daniel Bikel, Richard Schwartz, and Ralph Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning* 34, 1–3 (1999), 211–231.

Vinayak Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. 2001. Automatic segmentation of text into structured records. *SIGMOD Record* 30, 2 (2001), 175–186.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *Comput. Surveys* 41, 3 (2009), 15:1–15:58.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2012. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2012), 823–839.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *TIST* 2, 3 (2011), 27.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech andLanguage* 13, 4 (1999), 359–393.

Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Berlin.

Peter Christen. 2014. *Preparation of a real voter data set for record linkage and duplicate detection research*. Technical Report. Research School of Computer Science, The Australian National University.

Lukasz Ciszak. 2008. Application of clustering and association methods in data cleaning. In *IMCSIT*. IEEE, Wisla, 97–103.

Kaustav Das and Jeff Schneider. 2007. Detecting anomalous records in categorical datasets. In *SIGKDD*. ACM, San Jose, California, 220–229.

Kaustav Das, Jeff Schneider, and Daniel B Neill. 2008. Anomaly pattern detection in categorical datasets. In *SIGKDD*. ACM, Las Vegas, 169–176.

Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. 2011. CerFix: a system for cleaning data with certain fixes. In *VLDB*, Vol. 4. Seattle.

William A Gale and Geoffrey Sampson. 1995. Good-Turing frequency estimation without tears*. *Journal of Quantitative Linguistics* 2, 3 (1995), 217–237.

David Guthrie, Louise Guthrie, Ben Allison, and Yorick Wilks. 2007. Unsupervised Anomaly Detection. In *IJCAI*. Hyderabad, India, 1624–1628.

Victoria J Hodge and Jim Austin. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22, 2 (2004), 85–126.

Daniel Jurafsky and James H Martin. 2008. *Speech and Language Processing* (2nd ed.). Prentice Hall, Englewood Cliffs.

Michail Kazimianec and Nikolaus Augsten. 2011. PG-skip: Proximity graph based clustering of long strings. In *DASFAA*. Hong Kong, 31–46.

Shehroz S Khan and Michael G Madden. 2010. A survey of recent trends in one class classification. In *AICS, LNCS 6206*. Springer, Dublin, 188–197.

Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and HV Jagadish. 2008. Regular expression learning for information extraction. In *EMNLP*. ACM, Stroudsburg, 21–30.

Yunyao Li, Frederick R Reiss, and Laura Chiticariu. 2011. SystemT: A declarative information extraction system. In *ACL-HLT Systems Demonstrations*. Portland, 109–114.

Bin Liu, Laura Chiticariu, Vivian Chu, HV Jagadish, and Frederick R Reiss. 2010. Automatic rule refinement for information extraction. *PVLDB* 3, 1–2 (2010), 588–597.

Larry M Manevitz and Malik Yousef. 2002. One-class SVMs for document classification. *Journal of Machine Learning Research* 2 (2002), 139–154.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Arturas Mazeika and Michael Böhlen. 2006. Cleansing Databases of Misspelled Proper Nouns. In *CleanDB*. VLDB, Seoul.

Edward McFowland, Skyler Speakman, and Daniel B Neill. 2013. Fast generalized subset scan for anomalous pattern detection. *The Journal of Machine Learning Research* 14, 1 (2013), 1533–1561.

Thomas Müller and Hinrich Schütze. 2011. Improved modeling of out-of-vocabulary words using morphological classes. In *ACL*. Portland, 524–528.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research* 12 (2011), 2825–2830.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine Learning* 1, 1 (1986), 81–106.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation* 13, 7 (2001), 1443–1471.

Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. 2000. New support vector algorithms. *Neural Computation* 12, 5 (2000), 1207–1245.

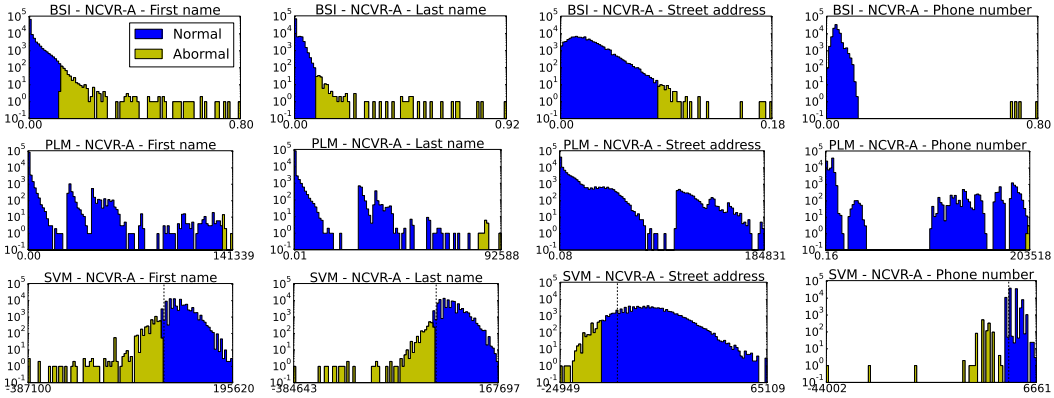
Hongzhi Wang, Mingda Li, Yingyi Bu, Jianzhong Li, Hong Gao, and Jiacheng Zhang. 2014. Cleanix: A Big Data Cleaning Parfait. In *CIKM*. ACM, Shanghai, 2024–2026.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1999. *Managing Gigabytes* (2nd ed.). Morgan Kaufmann, San Francisco.

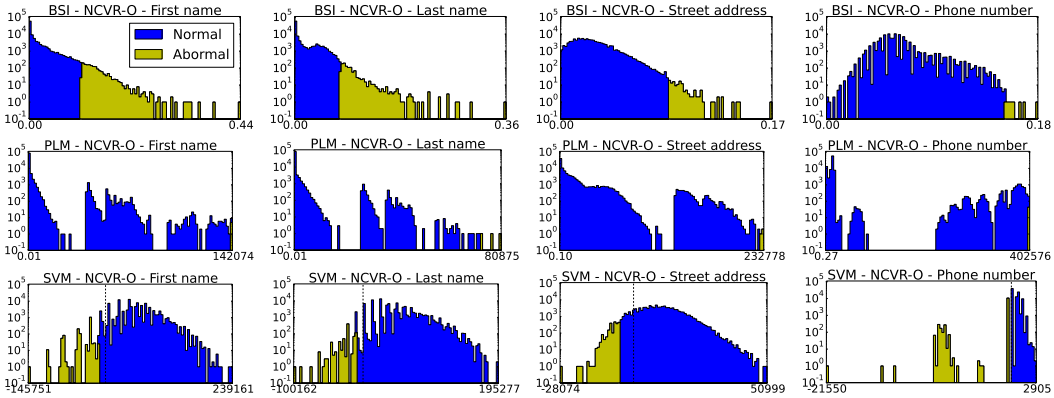
Shu Wu and Shengrui Wang. 2013. Information-theoretic outlier detection for large-scale categorical data. *IEEE Transactions on Knowledge and Data Engineering* 25, 3 (2013), 589–602.

Haiqin Yang, Irwin King, and Michael R Lyu. 2010. Multi-task learning for one-class classification. In *WCCI*. IEEE, Barcelona, 2820–2827.

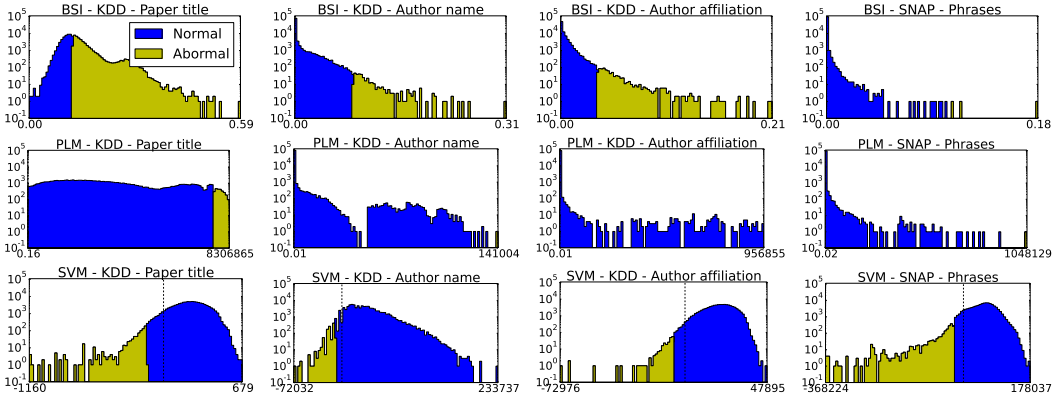
Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive Parser-Centric Text Normalization. In *ACL*. Sofia, 1159–1168.



(a) Distribution of normal and abnormal values for the NCVR April data set.



(b) Distribution of normal and abnormal values for the NCVR October data set.



(c) Distribution of normal and abnormal values for the KDD Cup (left three columns) and the SNAP memetracker (right-most column) data sets.

Fig. 6: Distribution of normal and abnormal values for the four data sets, averaged over all parameter settings for each technique and attribute. As described in Section 8.4, 'normal' and 'abnormal' are set according to the majority classification for each value. For **SVM** the vertical dotted lines show the decision boundary. Note the y-axes are on a logarithmic scale.

Table VII: Selected highly ranked abnormal (top three) and normal (bottom three) values in the NCVR-A (April 2014) data set as identified by the three techniques, and averaged over all parameter settings. For presentation reasons some values have been truncated.

First name	Last name	Street address	Phone number
BSI			
”we18” 82”n24 #894 w ”46un2	w160””d13 #2acad st	3‘ fernwood ct	h7216 ca” ” 10/23/2009”
tina /lea	g1 16,t ” ”	858 rural rte 3 (pct 11)	g208 #a dem
raphael(ray)	,florem6	349 rural rte 1 (pct 6)	” t #2”
aacia	aarhus	0 beach rd	252 633 4723
aadilah	aaberg	0 curtis creek rd	252 638 6211
aalima	aarne	0 lee ln	828 254 6211
PLM			
062”	2020999844	0 p.o. box 24	” t #2”
o”	27834c	793 rural rte 1 (pct 6)	h7216 ca” 47”apmi
”we18” 82”n24 #894 w ”46un2	””d222	0 rt 7 bx 421	g208 #a dem
erine	harrison	0 ncsu tucker	804 999 9999
linda	jones	0 ncsu watauga	828 000 0006
michael	roberson	101 s church st	828 298 3300
SVM			
zoe’	yu lay	509 m & m ln	g208 #a dem
””””6 2	w0gaman	41 p g ln	938 3567
za’heem	h15	19 r.d. trl nw	849 260 0
aacia	aaberg	10 apaloosa pl	336 111 1111
aaisha	adnesen	7 atkins st	336 447 4444
kirstin	agard	2 big bluff pl	518 999 9999

Table VIII: Selected highly ranked abnormal (top three) and normal (bottom three) values in the NCVR-O (October 2014) data set as identified by the three techniques, and averaged over all parameter settings. For presentation reasons some values have been truncated.

First name	Last name	Street address	Phone number
BSI			
co0k	sallie (pat)	36 goldenleaf (wolf laurel) rd	919 9)9 6823
jonathan2242	b0gli	250 creekside (mountain air) way #c-101	919 9)9 3396
r0se	;pgeman	605 jones ferry rd #hh06	919 909 (396
aaleah	aabid	0 alston st	0000
adam	aassar	3 n church st	252
alece	amir	12 ncsu carroll	828 298 9952
PLM			
j q	ujj	0 rr 3 box 457-o	401 741 8576
w f	qaq	0 po bx 369	601 441 8538
ga0	co0k	0 rt 4	992 290 3
anna	harrison	0 n west st	252
anne	anderson	0 gardner dorm	0000
nathanie	miller	0 ncsu carroll	334 999 9999
SVM			
zu	co0k	9192 nc 42	992 290 3
ga0	wo0dard	79 k-9 dr	919 9)9 6823
al-r	d’ugo	5916	252
aadam	aabrams	10 atlantic ave	260 999 9999
aaiysa	aalborg	11 allman hill rd	413 999 9999
aida	alders	10 ainsworth ct	770 530 7777

Received June 2015; revised xx; accepted yy

Table IX: Selected highly ranked abnormal (top three) and normal (bottom row) values in the KDD data set as identified by the three techniques, and averaged over all parameter settings. For presentation reasons some values have been truncated.

Author name	Paper title	Author affiliation
BSI		
E.t.s.e	4.35	http://www.ma.huji.ac.il/˜drorbn
FGAN / FHR-EL	P4-354	{lromdhan,
H. M1Rsal	Euporean Community Euro 0.4394 0.4335 0.4572	ABCDE; FGHIJ; KLMNO
A A Abrikosov	Comments and Discussion	ABB Corporate Research
A A Chaudhry	From the Chairman	AG
A A Natale	Letter from the Editor-in-Chief	ARC Centre of Excellence
PLM		
E.t.s.e	CVC4	QMW
GU	$:HEEDVHGSHUVRQDODQGSURIVVLRQDOGHPH	2; 2; 3
SWE-BRCA	TBL19-1	SHFJ
Thomas	Comments and Discussion	National University
Zheng Zhang	Editorial Note	Institute of Science
M. Chen	Guest editorial	Institute and State University
SVM		
h-z. hu	Y2K	p.147-169 . ISBN 3-540-5649-6
W. B.	The CH3 3pz 2A2	U.K.)
J. k. o. Sin	Supernovae 2006hu, 2006hv, 2006hw, 2006hx	Q.
A Agarwal	Abducing Temporal Discourse	University of California
A Ahlberg	Adaptive Dictionary Matching	Department of Statistics
A Battle	Bayesian Anomaly Detection	Laboratory of Neurogenetics

Table X: Selected highly ranked abnormal (top three) and normal (bottom three) values in the SNAP Memetracker data set as identified by the three techniques, and averaged over all parameter settings. For presentation reasons some values have been truncated.

Phrases
BSI
0 0 ytqxzwiwmjk5ztlmnmvim0odkz5f3t7ax1zx6tnxwo7 created directory fglrx-install ou6191 verifying tag blogger com 1999 blog-31636059 post-3432464819012682047
0 0 if you are not on the default display you may also need to set the permission 0 database username password 02 02 2009
PLM
0 0 ytqxzwiwmjk5ztlmnmvim0odkz5f3t7ax1zx6tnxwo7 56 4d f7 d7 cb d8 c9 be-26 8e d9 7c 2f 04 a9 8c 1e 2f 3g 4h 5i
things to see soon in case it dies before you do and select it then you send a message saying she really needs to gain a lot
SVM
wild updated 2009-03-31 16 44 26 gmt w w l h v b y a e a k n o t p f h r s o t e e t r f l a t v t i t e e o d a f f 0 8 h 1 9 4 7
1 1 percent of the total audit-related contacts planned for the year we have designed and built our property with special attention to energy efficiency we are pleased to have linde as a partner of tongji university for this pioneering project